comment installer-umami-analytics sur-debian-12

Umami is a very lightweight, open-source, self-hosted web analytics solution. It is a good privacy-focused alternative to Google Analytics and other paid analytic solutions. One of the main advantages of using Umami is that it doesn't place any cookie on the user's browser, which means you don't need to put up the annoying cookie banner on your website.

In this tutorial, we will learn how to install Umami analytics on a Debian 12 server and use it to track websites.

Prerequisites

- A server running Debian 12.
- A non-root user with sudo privileges.
- A fully qualified domain name (FQDN) like `umami.example.com` pointing to the server.
- The Uncomplicated Firewall(UFW) is enabled and running.
- Update everything.

```
$ sudo apt update && sudo apt upgrade
```

- Install essential packages that your system needs. Some of these packages may already be installed on your system.

```
$ sudo apt install wget curl nano ufw software-properties-common dirmngr apt-transport-https gnupg2 ca-certificates lsb-release debian-archive-keyring unzip -y
```

Step 1 - Configure Firewall

The first step is to configure the firewall. Ubuntu comes with ufw (Uncomplicated Firewall) by default.

Check if the firewall is running.

```
$ sudo ufw status
```

You should get the following output.

```
Status: inactive
```

Allow SSH port so that the firewall doesn't break the current connection on enabling it.

```
$ sudo ufw allow OpenSSH
```

Allow HTTP and HTTPS ports as well.

```
$ sudo ufw allow http
$ sudo ufw allow https
```

Enable the Firewall

```
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

Check the status of the firewall again.

```
$ sudo ufw status
```

You should see a similar output.

```
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443                        ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
```

Step 2 - Install Git

Git is needed to clone Umami's official repository. Install Git.

```
$ sudo apt install git
```

Verify the installation.

```
$ git --version
git version 2.39.2
```

Set initial configuration variables.

```
$ git config --global user.name "Your Name"
$ git config --global user.email "email@example.com"
```

Step 3 - Install Node

Umami is a JavaScript app that runs on Nodejs. To install Node, we will use Nodesource's installer. Since Node v16.0 is the current stable version, we will install that.

Download and import the Nodesource GPG key

```
$ curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key | sudo gpg --dearmor -o /usr/share/keyrings/nodesource.gpg
```

Create Node Deb repository.

```
$ NODE_MAJOR=18
$ echo "deb [signed-by=/usr/share/keyrings/nodesource.gpg] https://deb.nodesource.com/node_$NODE_MAJOR.x nodistro main" | sudo tee /etc/apt/sources.list.d/nodesource.list
```

Update the Debian system package repository list.

```
$ sudo apt update
```

Install Node.

```
$ sudo apt install nodejs
```

Verify the Node installation.

```
$ node --version
v18.18.0
```

Step 4 - Install MariaDB Server

Debian 12 does not ship with MySQL by default and they haven't released an official package for it yet. Therefore, we will be using MariaDB for it.

Debian 12 ships with MariaDB 10.11.4 We will install that. You can however install the latest version from the repository.

```
$ sudo apt install mariadb-server
```

Check the version of MySQL.

```
$ mysql --version
mysql  Ver 15.1 Distrib 10.11.4-MariaDB, for debian-linux-gnu (x86_64) using  EditLine wrapper
```

Run the MariaDB secure install script.

```
$ sudo mariadb-secure-installation
```

You will be asked for the root password. Press **Enter** because we haven't set any password for it.

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
```

Next, you will be asked if you want to switch to the Unix socket authentication method. The `unix_socket` plugin allows you to use your operating system credentials to connect to the MariaDB server. Since you already have a protected root account, enter `n` to proceed.

```
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
```

Next, you will be asked if you want to change your root password. On Debian 12, the root password is tied closely to automated system maintenance, so it should be left alone. Type `n` to proceed further.

```
 ... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] n
```

Next, you will be asked certain questions to improve MariaDB security. Type **Y** to remove anonymous users, disallow remote root logins, remove the test database, and reload the privilege tables.

```
 ... skipping.

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

You can enter the MariaDB shell by typing `sudo mysql` or `sudo mariadb` on the command line.

Step 5 - Download Umami

The first step is to install the Yarn package manager. We can install it using NPM.

```
$ sudo npm install -g yarn
```

Since Umami is a Node application and doesn't have a public webroot directory, we don't need to host it via `/var/www` directory.

Clone the Umami's GitHub repository.

```
$ git clone https://github.com/mikecao/umami.git
```

Switch to the newly created directory.

```
$ cd umami
```

Install the Umami modules.

```
$ yarn install
```

Step 6 - Configure Umami

Create MySQL Credentials and populate the database

Enter the MySQL shell. Enter your root password to continue.

```
$ sudo mysql
```

Create `umami` user. Make sure the password meets the requirements set before.

```
mysql> CREATE USER 'umamiuser'@'localhost' IDENTIFIED BY 'YourPassword';
```

Create `umami` database.

```
mysql> CREATE DATABASE umami;
```

Grant the user privileges on the `umami` database.

```
mysql> GRANT ALL PRIVILEGES ON umami.* TO 'umamiuser'@'localhost';
```

Since we are not modifying the root user, you should create another SQL user for performing administrative tasks that employ password authentication. Choose a strong password for this one.

```
MariaDB> GRANT ALL ON *.* TO 'navjot'@'localhost' IDENTIFIED BY 'Yourpassword32!' WITH GRANT OPTION;
```

Flush privileges.

```
mysql> FLUSH PRIVILEGES;
```

Exit the Shell.

```
mysql> exit
```

Configure Umami Environment Variables

We need a strong App secret for logging purposes. For this, we will use the OpenSSL command.

```
$ openssl rand 30 | openssl base64 -A
bu4orqfdlG+Dh3Otau4oWSBbI9kGWSkGfYc/WiH/
```

Create a `.env` file to store environment variables for Umami's installation.

```
$ touch .env
```

Open the file for editing.

```
$ nano .env
```

Paste the following code in it. You will need to encode any special characters in your password for the database URL. Use the meyerweb encoder for it. In our case, the `#` is translated to `%23`. The database URL ends with the database name we need to connect to. Use the app secret generated earlier for the `APP_SECRET` variable. The `DISABLE_TELEMETRY=1` option disables sending anonymous data by the app to Umami's servers. The `TRACKER_SCRIPT_NAME` variable is useful to avoid getting your script blocked by Ad blockers. Give it a unique name something that is unique to your website.

```
DATABASE_URL=mysql://umamiuser:YourPassword@localhost:3306/umami
APP_SECRET=bu4orqfdlG+Dh3Otau4oWSBbI9kGWSkGfYc/WiH/
DISABLE_TELEMETRY=1
TRACKER_SCRIPT_NAME=custom
```

Step 7 - Running Umami

Now that everything is set up, build the Umami application.

```
$ yarn build
```

The next step is to start the application. We can start the app by using the command `yarn start` but it would mean that you need to keep the terminal open for Umami to run. Therefore, we need a way to run Umami in the background. To do this, we will install PM2 (Advanced Production Process Manager for Node).

Install PM2.

```
$ sudo yarn global add pm2
```

The `global` options means we are installing PM2 globally, and therefore, we need **sudo** privileges to run the command.

Start the Umami application.

```
$ pm2 start yarn --name umami -- start
```

You will get the following output.

```
[PM2] Starting /usr/bin/yarn in fork_mode (1 instance)
[PM2] Done.
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
? id ? name      ? namespace   ? version ? mode    ? pid      ? uptime ? ?      ? status    ? cpu      ? mem       ? user     ? watching ?
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
? 0  ? umami     ? default     ? N/A     ? fork    ? 2020     ? 0s     ? 0    ? online    ? 0%       ? 18.8mb    ? navjot   ? disabled ?
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
```

Save the Umami application with PM2 for further use.

```
$ pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /home/navjot/.pm2/dump.pm2
```

Umami will automatically restart if it crashes or is killed but not if the system is rebooted. We need to create a `systemd` script to ensure it restarts across system reboots. Run the following command to generate a startup script.

```
$ pm2 startup
```

The resulting output will give you a command to set PM2 to run on boot. The output, in your case, will give you the current username. Run the following command to generate the startup script.

```
$ sudo env PATH=$PATH:/usr/bin /usr/local/share/.config/yarn/global/node_modules/pm2/bin/pm2 startup systemd -u navjot --hp /home/navjot
```

Step 8 - Install Nginx

Debian 12 ships with an older version of Nginx. To install the latest version, you need to download the official Nginx repository.

Import Nginx's signing key.

```
$ curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor \
    | sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg >/dev/null
```

Add the repository for Nginx's stable version.

```
$ echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] \
http://nginx.org/packages/debian `lsb_release -cs` nginx" \
    | sudo tee /etc/apt/sources.list.d/nginx.list
```

Update the system repositories.

```
$ sudo apt update
```

Install Nginx.

```
$ sudo apt install nginx
```

Verify the installation. On Debian systems, the following command will only work with `sudo`.

```
$ sudo nginx -v
nginx version: nginx/1.24.0
```

Start Nginx.

```
$ sudo systemctl start nginx
```

Check the service status.

```
$ sudo systemctl status nginx
? nginx.service - nginx - high performance web server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Tue 2023-10-10 11:19:45 UTC; 9s ago
       Docs: https://nginx.org/en/docs/
    Process: 3646 ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf (code=exited, status=0/SUCCESS)
   Main PID: 3647 (nginx)
      Tasks: 3 (limit: 4652)
     Memory: 2.4M
        CPU: 8ms
     CGroup: /system.slice/nginx.service
             ??3647 "nginx: master process /usr/sbin/nginx -c /etc/nginx/nginx.conf"
             ??3648 "nginx: worker process"
             ??3649 "nginx: worker process"

Oct 10 11:19:45 umami systemd[1]: Starting nginx.service - nginx - high performance web server...
Oct 10 11:19:45 umami systemd[1]: Started nginx.service - nginx - high performance web server.
```

Step 9 - Install SSL using Let's Encrypt

We need to install Certbot to generate free SSL certificates offered by Let's Encrypt.

You can install Certbot using Debian's repository or grab the latest version using the Snapd tool. We will be using the Snapd version.

Debian 12 comes doesn't come with Snapd installed. Install Snapd package.

```
$ sudo apt install snapd
```

Ensure that your version of Snapd is up to date.

```
$ sudo snap install core
$ sudo snap refresh core
```

Install Certbot.

```
$ sudo snap install --classic certbot
```

Use the following command to ensure that the Certbot command can be run by creating a symbolic link to the `/usr/bin` directory.

```
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

Verify the installation.

```
$ certbot --version
certbot 2.7.0
```

Generate the SSL certificate.

```
$ sudo certbot certonly --nginx --agree-tos --no-eff-email --staple-ocsp --preferred-challenges http -m name@example.com -d umami.example.com
```

The above command will download a certificate to the `/etc/letsencrypt/live/umami.example.com` directory on your server.

Generate a **Diffie-Hellman group** certificate.

```
$ sudo openssl dhparam -dsaparam -out /etc/ssl/certs/dhparam.pem 4096
```

Check the Certbot renewal scheduler service.

```
$ sudo systemctl list-timers
```

You will find `snap.certbot.renew.service` as one of the services scheduled to run.

```
NEXT                        LEFT        LAST                        PASSED       UNIT                         ACTIVATES
.....
Tue 2023-10-10 14:24:00 UTC 1h 55min left -                           -           snap.certbot.renew.timer     snap.certbot.renew.service
Wed 2023-10-11 00:00:00 UTC 11h left      -                           -           dpkg-db-backup.timer         dpkg-db-backup.service
Wed 2023-10-11 00:00:00 UTC 11h left      Tue 2023-10-10 00:00:04 UTC 12h ago     exim4-base.timer             exim4-base.service
```

Do a dry run of the process to check whether the SSL renewal is working fine.

```
$ sudo certbot renew --dry-run
```

If you see no errors, you are all set. Your certificate will renew automatically.

Step 10 - Configure Nginx

Create and open the file `/etc/nginx/conf.d/umami.conf` for editing.

```
$ sudo nano /etc/nginx/conf.d/umami.conf
```

Paste the following code in it.

```
server {
    listen       443 ssl http2;
    listen       [::]:443 ssl http2;
    server_name  umami.example.com;

    access_log  /var/log/nginx/umami.access.log;
    error_log   /var/log/nginx/umami.error.log;

    # SSL
    ssl_certificate      /etc/letsencrypt/live/umami.example.com/fullchain.pem;
    ssl_certificate_key  /etc/letsencrypt/live/umami.example.com/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/umami.example.com/chain.pem;
    ssl_session_timeout  5m;
    ssl_session_cache shared:MozSSL:10m;
    ssl_session_tickets off;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DH
    ssl_ecdh_curve X25519:prime256v1:secp384r1:secp521r1;
    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;
    resolver 8.8.8.8;

    location / {
```

```
        proxy_pass http://localhost:3000;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
# enforce HTTPS
server {
    listen       80;
    listen       [::]:80;
    server_name  umami.example.com;
    return 301   https://$host$request_uri;
}
```

Save the file by pressing **Ctrl + X** and entering **Y** when prompted once finished.

Open the file `/etc/nginx/nginx.conf` for editing.

```
$ sudo nano /etc/nginx/nginx.conf
```

Add the following line before the line `include /etc/nginx/conf.d/*.conf;`.

```
server_names_hash_bucket_size  64;
```

Save the file by pressing **Ctrl + X** and entering **Y** when prompted.

Verify the Nginx configuration file syntax.

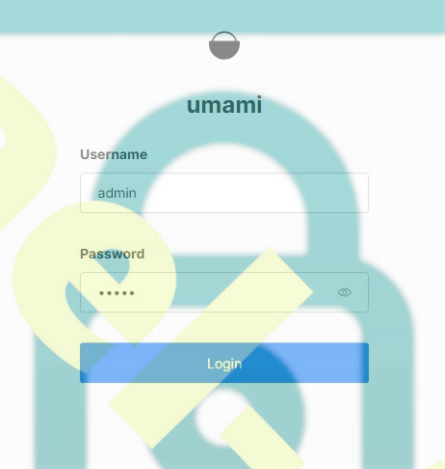
```
$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Restart the Nginx service to enable the new configuration.

```
$ sudo systemctl restart nginx
```

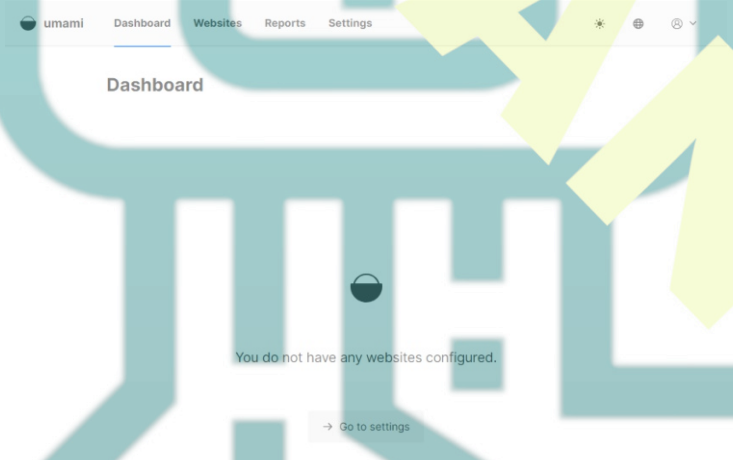
Step 11 - Set up a Site and Collect Statistics

Open the URL `https://umami.example.com` in your browser, and it will look like the following.

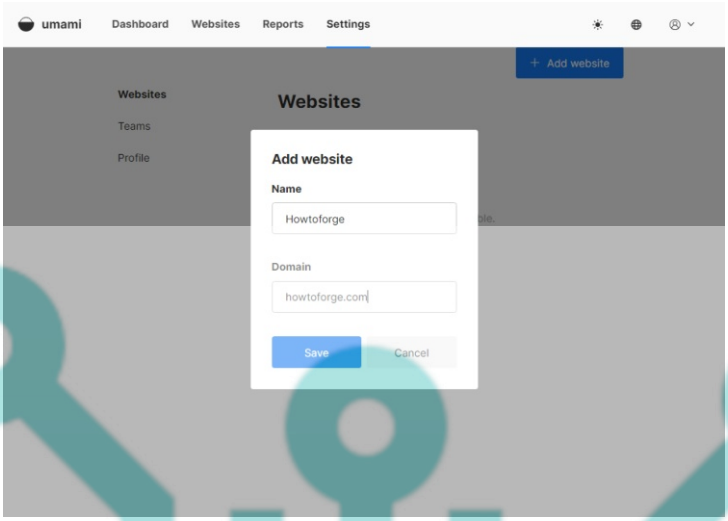


Umami generates a default administrator account with the username **admin** and password as **umami** during the installation. Use these credentials to log in.

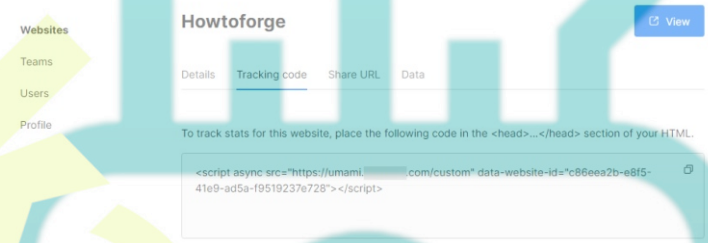
Once logged in, you will be greeted with the following page.



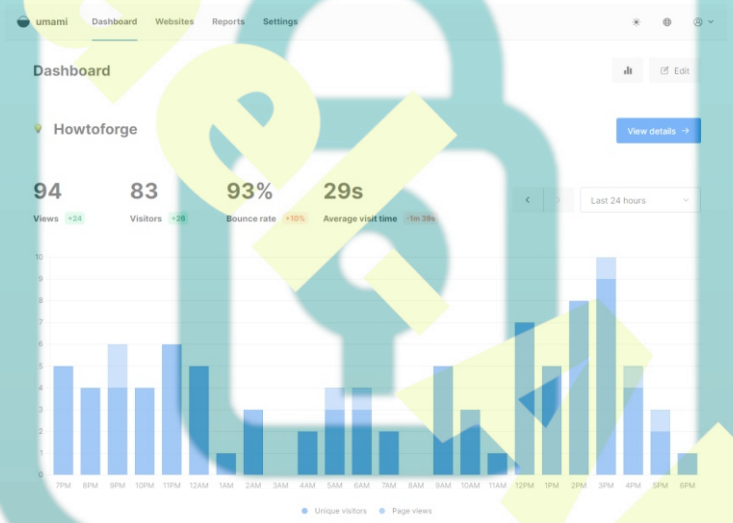
Visit the settings page and click on **Add website** to get started.



Once you have added the site, click on the **Edit** button and then switch to the **Tracking code** tab to get the code.



Paste the code between your header or footer HTML tags, and within a few minutes, Umami will start getting data.



Step 12 - Update Umami

To update Umami, shift to the Umami installation directory.

```
$ cd ~/umami
```

Stop Umami application.

```
$ pm2 stop umami
```

Install any new or updated dependencies.

```
$ yarn install
```

Rebuild the Umami application.

```
$ yarn build
```

Start Umami application.

```
$ pm2 start umami
```

Conclusion

This concludes our tutorial on installing and setting up the Umami Analytics tool on a Debian 12 server. If you have any questions, post them in the comments below.